Noisebridge Neural Networks

Christoph Maier

11 March 2009

Neuron

- **General neuron: nonlinear function of scalar product**

```
neuron = Function[{input, weights, sigmoid}, sigmoid[weights.input]]
```

```
Function[{input, weights, sigmoid}, sigmoid[weights.input]]
```

- **Unit step neuron**

```
stepneuron =
 Function[{input, weights, threshold}, neuron[input, weights, UnitStep[# - threshold] &]]
```

```
Function[{input, weights, threshold}, neuron[input, weights, UnitStep[#1 - threshold] &]]
```

Learning

```
educate = Function[{input, target, nerve, weights, tweak, rate},
  {weights + rate input (target - nerve[input, weights]), tweak[rate]}]
```

```
Function[{input, target, nerve, weights, tweak, rate},
 {weights + rate input (target - nerve[input, weights]), tweak[rate]}]
```

Invariants under learning iteration: type of neuron, Function how to tweak learning rate [, log]
Iterated: weight vector, learning rate
Variant inputs: Learning vector, learning target

```
educatiter = Function[{invariant, iterated, variant},
  Block[
   {nerve = invariant[[1]], ratetweak = invariant[[2]],
    input = variant[[1]], target = variant[[2]],
    weights = iterated[[1]], rate = iterated[[2]],
    output},
   output = educate[input, target, nerve, weights, ratetweak, rate]
  ]
 ]
```

```
Function[{invariant, iterated, variant},
 Block[{nerve = invariant[[1]], ratetweak = invariant[[2]], input = variant[[1]],
   target = variant[[2]], weights = iterated[[1]], rate = iterated[[2]], output},
  output = educate[input, target, nerve, weights, ratetweak, rate]]]
```

Training set

```
trainingset = {{{1, 0, 0}, 1}, {{1, 0, 1}, 1}, {{1, 1, 0}, 1}, {{1, 1, 1}, 0}}
```

```
{{{1, 0, 0}, 1}, {{1, 0, 1}, 1}, {{1, 1, 0}, 1}, {{1, 1, 1}, 0}}
```

Test set

```
testset = {{0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {0, 1, 1}, {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}}
```

```
{{0, 0, 0}, {0, 0, 1}, {0, 1, 0}, {0, 1, 1}, {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}}
```

Learning

- **Single learning step**

```
educate[{1, 0, 0}, 1, stepneuron[#1, #2, 1/2] &, {0, 0, 0}, .95 # &, .8]
```

```
{{0.8, 0, 0}, 0.76}
```

- **Single learning step with iteratable parameters**

```
educatiter[{stepneuron[#1, #2, 1/2] &, .95 # &}, {{0, 0, 0}, .8}, {{1, 0, 0}, 1}]
```

```
{{0.8, 0, 0}, 0.76}
```

- **Run once over the training set**

```
FoldList[educatiter[{stepneuron[#1, #2, 1/2] &, .99 # &}, #1, #2] &, {{0, 0, 0}, .8}, trainingset]
```

```
{{{0, 0, 0}, 0.8}, {{0.8, 0, 0}, 0.792}, {{0.8, 0, 0}, 0.78408},
 {{0.8, 0, 0}, 0.776239}, {{0.0237608, -0.776239, -0.776239}, 0.768477}}
```

Run 10 times over the training set

```
NestList[
  FoldList[educatiter[{stepneuron[#1, #2, 1/2] &, .99 # &}, #1, #2] &, Last[#], trainingset] &,
  {{{0, 0, 0}, .8}}, 10];
run10 = Join @@ (Rest /@ %)
```

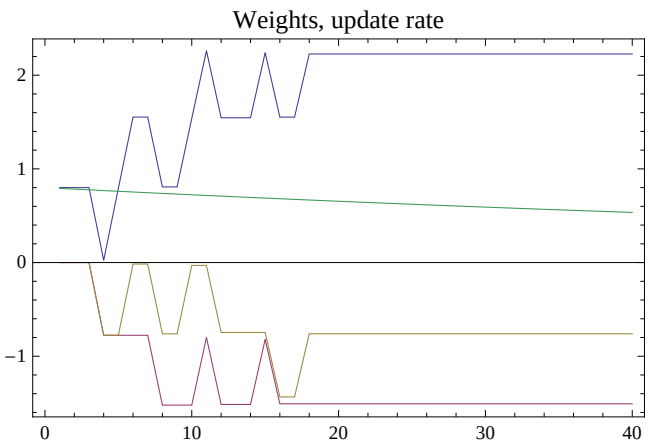
```
{{{0.8, 0, 0}, 0.792}, {{0.8, 0, 0}, 0.78408},
 {{0.8, 0, 0}, 0.776239}, {{0.0237608, -0.776239, -0.776239}, 0.768477},
 {{0.792238, -0.776239, -0.776239}, 0.760792}, {{1.55303, -0.776239, -0.0154472}, 0.753184},
 {{1.55303, -0.776239, -0.0154472}, 0.745652}, {{0.807377, -1.52189, -0.761099}, 0.738196},
 {{0.807377, -1.52189, -0.761099}, 0.730814}, {{1.53819, -1.52189, -0.0302856}, 0.723506},
 {{2.2617, -0.798386, -0.0302856}, 0.716271}, {{1.54543, -1.51466, -0.746556}, 0.709108},
 {{1.54543, -1.51466, -0.746556}, 0.702017}, {{1.54543, -1.51466, -0.746556}, 0.694997},
 {{2.24042, -0.81966, -0.746556}, 0.688047}, {{1.55238, -1.50771, -1.4346}, 0.681166},
 {{1.55238, -1.50771, -1.4346}, 0.674355}, {{2.22673, -1.50771, -0.760248}, 0.667611},
 {{2.22673, -1.50771, -0.760248}, 0.660935}, {{2.22673, -1.50771, -0.760248}, 0.654326},
 {{2.22673, -1.50771, -0.760248}, 0.647782}, {{2.22673, -1.50771, -0.760248}, 0.641304},
 {{2.22673, -1.50771, -0.760248}, 0.634891}, {{2.22673, -1.50771, -0.760248}, 0.628543},
 {{2.22673, -1.50771, -0.760248}, 0.622257}, {{2.22673, -1.50771, -0.760248}, 0.616035},
 {{2.22673, -1.50771, -0.760248}, 0.609874}, {{2.22673, -1.50771, -0.760248}, 0.603775},
 {{2.22673, -1.50771, -0.760248}, 0.597738}, {{2.22673, -1.50771, -0.760248}, 0.59176},
 {{2.22673, -1.50771, -0.760248}, 0.585843}, {{2.22673, -1.50771, -0.760248}, 0.579984},
 {{2.22673, -1.50771, -0.760248}, 0.574184}, {{2.22673, -1.50771, -0.760248}, 0.568443},
 {{2.22673, -1.50771, -0.760248}, 0.562758}, {{2.22673, -1.50771, -0.760248}, 0.557131},
 {{2.22673, -1.50771, -0.760248}, 0.551559}, {{2.22673, -1.50771, -0.760248}, 0.546044},
 {{2.22673, -1.50771, -0.760248}, 0.540583}, {{2.22673, -1.50771, -0.760248}, 0.535177}}
```

Sanity check

```
stepneuron[#1, First[Last[run10]], 1/2] & /@ (First /@ trainingset)
```

```
{1, 1, 1, 0}
```

```
ListPlot[Transpose[Flatten /@ run10], Frame → True,
 Joined → True, PlotLabel → "Weights, update rate"]
```

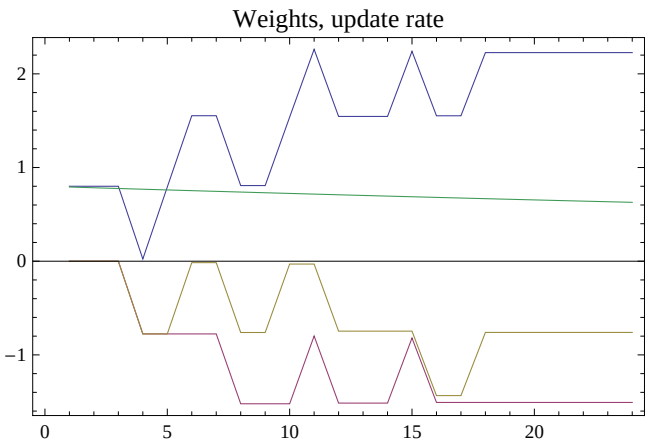


■ Terminate automatically

```
NestWhileList[
  FoldList[educatiter[{stepneuron[#1, #2, 1/2] &, .99 # &}, #1, #2] &, Last[#], trainingset] &,
  {{{0, 0, 0}, .8}}, First[Last[#1]] ≠ First[Last[#2]] &, 2];
autorun = Join @@ (Rest /@ %)
```

```
{{{0.8, 0, 0}, 0.792}, {{0.8, 0, 0}, 0.78408},
 {{0.8, 0, 0}, 0.776239}, {{0.0237608, -0.776239, -0.776239}, 0.768477},
 {{0.792238, -0.776239, -0.776239}, 0.760792}, {{1.55303, -0.776239, -0.0154472}, 0.753184},
 {{1.55303, -0.776239, -0.0154472}, 0.745652}, {{0.807377, -1.52189, -0.761099}, 0.738196},
 {{0.807377, -1.52189, -0.761099}, 0.730814}, {{1.53819, -1.52189, -0.0302856}, 0.723506},
 {{2.2617, -0.798386, -0.0302856}, 0.716271}, {{1.54543, -1.51466, -0.746556}, 0.709108},
 {{1.54543, -1.51466, -0.746556}, 0.702017}, {{1.54543, -1.51466, -0.746556}, 0.694997},
 {{2.24042, -0.81966, -0.746556}, 0.688047}, {{1.55238, -1.50771, -1.4346}, 0.681166},
 {{1.55238, -1.50771, -1.4346}, 0.674355}, {{2.22673, -1.50771, -0.760248}, 0.667611},
 {{2.22673, -1.50771, -0.760248}, 0.660935}, {{2.22673, -1.50771, -0.760248}, 0.654326},
 {{2.22673, -1.50771, -0.760248}, 0.647782}, {{2.22673, -1.50771, -0.760248}, 0.641304},
 {{2.22673, -1.50771, -0.760248}, 0.634891}, {{2.22673, -1.50771, -0.760248}, 0.628543}}
```

```
ListPlot[Transpose[Flatten /@ autorun], Frame → True,
 Joined → True, PlotLabel → "Weights, update rate"]
```



■ Sanity check

```
stepneuron[#1, First[Last[autorun]], 1/2] & /@ (First /@ trainingset)
```

```
{1, 1, 1, 0}
```

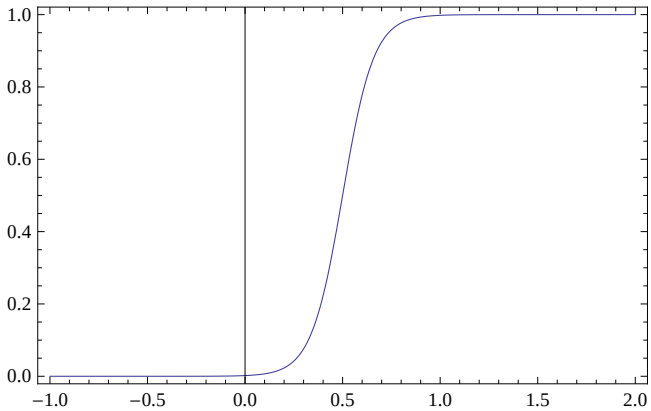
```
stepneuron[#1, First[Last[autorun]], 1/2] & /@ testset
```

```
{0, 0, 0, 0, 1, 1, 1, 0}
```

17 March 2009

New sigmoid function

```
Plot[1/(1 + Exp[(ϵ - μ)/kT]) /. {μ → .5, kT → -.08}, {ϵ, -1, 2}, Frame → True]
```



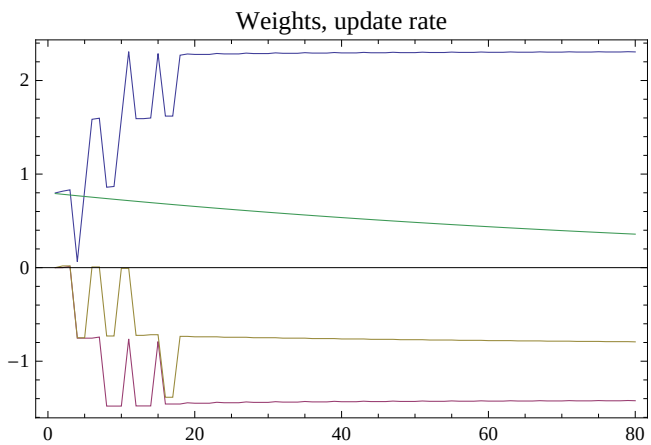

```
fermineuron = Function[{input, weights, μ, kT}, neuron[input, weights, 1/(1 + Exp[(# - μ)/kT]) &]]
```

```
Function[{input, weights, μ, kT}, neuron[input, weights, 1/(1 + Exp[(#1 - μ)/kT]) &]]
```

- **Run 20 times over the training set**

```
NestList[FoldList[educatiter[{fermineuron[#1, #2, .5, -.08] &, .99 # &}, #1, #2] &,
    Last[#], trainingset] &, {{{0, 0, 0}, .8}}, 20];
fermirun20 = Join @@ (Rest /@ %);

ListPlot[Transpose[Flatten /@ fermirun20],
 Frame → True, Joined → True, PlotLabel → "Weights, update rate"]
```



- **Sanity check**

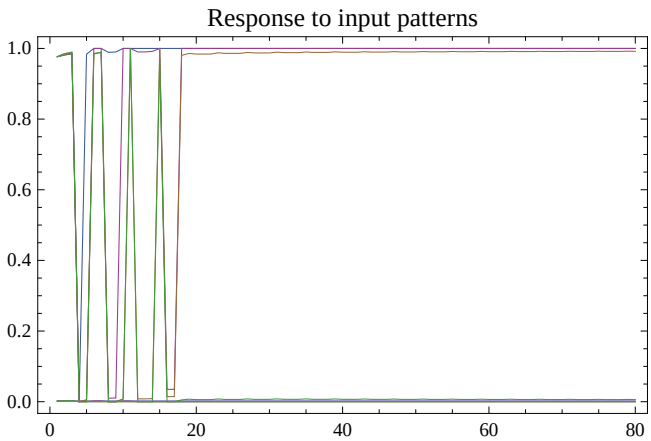
```
fermineuron[#1, First[Last[fermirun20]], .5, -.08] & /@ (First /@ trainingset)
```

```
{1., 0.999997, 0.991781, 0.00593264}
```

```
fermineuron[#1, First[Last[fermirun20]], .5, -.08] & /@ testset
```

$$\{0.00192673,\ 9.54815\times10^{-8},\ 3.66793\times10^{-11},\ 1.81418\times10^{-15},\ 1.,\ 0.999997,\ 0.991781,\ 0.00593264\}$$

```
ListPlot[Function[{pattern}, fermineuron[pattern, First[#], .5, -.08] & /@ fermirun20] /@
  testset, Frame → True, PlotLabel → "Response to input patterns", Joined → True]
```



NOR Training set

```
nortraining = {{{1, 0, 0}, 1}, {{1, 0, 1}, 0}, {{1, 1, 0}, 0}, {{1, 1, 1}, 0}}
```

```
{{{1, 0, 0}, 1}, {{1, 0, 1}, 0}, {{1, 1, 0}, 0}, {{1, 1, 1}, 0}}
```

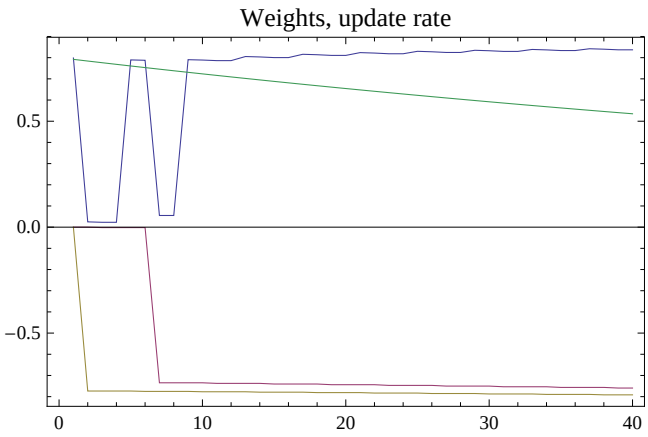
- **Run 10 times over the training set**

```
NestList[FoldList[educatiter[{fermineuron[#1, #2, .5, -.08] &, .99 # &}, #1, #2] &,
    Last[#], nortraining] &, {{{0, 0, 0}, .8}}, 10];
norrun10 = Join @@ (Rest /@ %);
```

```
Last[norrun10]
```

```
{{0.837052, -0.759411, -0.79151}, 0.535177}
```

```
ListPlot[Transpose[Flatten /@ norrun10],
 Frame → True, Joined → True, PlotLabel → "Weights, update rate"]
```



- **Sanity check**

```
fermineuron[#1, First[Last[norrun10]], .5, -.08] & /@ testset
```

$$\{0.00192673,\ 9.74555 \times 10^{-8},\ 1.45567 \times 10^{-7},\ 7.34867 \times 10^{-12},\ 0.985416,\ 0.00339952,\ 0.00506926,\ 2.57216 \times 10^{-7}\}$$

```
ListPlot[Function[{pattern}, fermineuron[pattern, First[#], .5, -.08] & /@ norrun10] /@ testset,
 Frame → True, PlotLabel → "Response to input patterns", PlotRange → All, Joined → True]
```